02

# Creando el modelo producto

## Transcripción

[00:00] Hola. Para avanzar con el curso y la evolución de nuestra aplicación, vamos a revisar el método guardar de la clase productoController. Estamos realizando la operación de insert en la tabla de producto, con los parámetros que recibimos en el método, y lo enviamos como parámetros de try(statement).

[00:18] Hasta ahora venimos tratando los atributos de las tablas como variables sueltas, sin ninguna representatividad del producto en el código, tal como tenemos en la tabla de la base de datos.

[00:29] La idea ahora es, en lugar de estar asignando variables sueltas, como estamos haciendo acá, recibidas como parámetros, vamos a crear una forma de representar el producto directamente en el proyecto en Java. O sea, vamos a crear un modelo y para eso, nosotros vamos a crear una clase en un nuevo paquete y esta clase va a ser llamada de producto, para representar justamente la tabla de producto.

[00:56] Entonces acá, en nuestro Explorer nosotros vamos a hacer un new de package y el paquete va a llamarse com.alura.jdbc.modelo. ¿Por qué? Van a ser las clases de modelo que representan las tablas, pero en la aplicación. Tenemos el paquete y aquí en el paquete vamos a hacer un new class que va a llamarse de producto. Perfecto.

[01:25] Ahí tenemos la clase producto, que va a empezar a representar la tabla de producto. Bueno, como una idea es que el producto, acá la clase, represente la tabla, entonces esta clase tiene que tener los mismo campos de la tabla. Lc

campos son entonces un private Integer id; vamos a tener también un private String descripcion; vamos a tener un private String nombre;

[01:55] Acá voy a cambiar el orden para que quede igual que la tabla. Por último un private Integer cantidad; ahí está. Todos estos atributos están como privados para que no tengamos el encapsulamiento de la clase.

[02.12] Por ahora, vamos a dejarlos así. Vamos a crear los métodos, los getters, setters, constructores acorde vayamos necesitando en el desarrollo del proyecto. Voy a guardar acá en la clase. Guardé el producto, dejé los campos acá y ahora volvemos para el productoController en el método guardar. Voy a agrandar una vez más acá, ahí está.

[02:35] Vamos a entrar al método que llama el guardar. O sea, para saberlo vamos a hacer un clic acá, en el método guardar, clic derecho del mouse y vamos a elegir esta opción de acá: Open Call Hierarchy. Hacemos un clic en ella y acá se abrió el método rojito acá, guardar de ControlDeStockFrame.

[02:58] De esa forma nosotros logramos saber quién está llamando a este método. Puede ser en más de un punto de la aplicación. En este caso tenemos un solo punto que es el método guardar del frame. En este método de guardar nosotros estamos tomando los datos del formulario y lo estamos agregando acá, adentro de un HashMap.

[03:19] Esto de acá, este HashMap lo estamos enviando al productoController para hacer el guardado allá, el insert en la tabla de producto. Esto nosotros lo vamos a cambiar. Entonces acá, este HashMap va a tornarse el producto ahora. Entonces vamos a cambiar el HashMap para el producto que recién creamos. Vamos a tomar estas variables que están acá, todas estas de producto de input, que tenemos de nombre, descripción y la cantidad entero.

[03:49] En lugar de estar transformando todo en string, vamos a hacer un new Producto() y adentro acá del constructor que aún no tenemos, nosotros vamos

a estar agregando nuestros parámetros para este constructor. Al final de esto, nosotros vamos a crearlo para asignar los valores a las variables.

[04:13] Como la cantidadInt ya está en el tipo que nosotros tenemos en la clase de producto, nosotros no necesitamos hacer la conversión de string.valueOf como estábamos haciendo. Entonces esta parte se borra, el producto acá, como el Eclipse se queja, nosotros hacemos un control o command 1 primero para importar la clase y ahora sí vamos a crear un constructor para este tipo producto.

[04:41] Lo voy a mover acá para dejar un poco más ordenado. Ahora sí. El primer campo es el nombre, el segundo es la descripción y el tercero es la cantidad. Esos campos van a ser un this.nombre = nombre; this.descripcion = descripcion; y this.cantidad = cantidad; ahí está.

[05:12] Inicializamos nuestro objeto y lo estamos enviando para el método guardar de productoController. Acá el método guardar queda acá rojito, una queja, porque nosotros estamos enviando un objeto del tipo producto y en la declaración del método, de la firma del método, estamos esperando un map del String string. Lo que vamos a hacer acá es cambiar esto para el producto también.

[05:38] Producto, ahí está. Importo la clase, guardo todo y cuando vuelvo para el frame ya no se queja más, está todo bien. Vayamos ahora entonces para el productoController porque acá tenemos que seguir haciendo unos cambios. Acá como cambiamos el map por el objeto del producto, ya no tenemos más cómo buscar las informaciones por la clave del HashMap.

[06:06] ¿Entonces qué vamos a hacer acá? El producto.get va a ser getNombre, vamos a crear un método para esto, getNombre. Lo mismo va a pasar para la descripción. Vamos a crear este método también, y ya no necesitamos más hacer este cambio, esta conversión en realidad, porque ya tenemos acá el getCantidad con el tipo correcto.

[06:35] Lo que no tenemos todavía es la declaración de estos métodos en la clase de producto. En lugar de salir creando una por vez acá, vengamos a la clase de producto y vamos a hacer lo siguiente. Vamos en el teclado a utilizar las siguientes teclas: La command o Control y la 3, número 3 y vamos a escribir ggas, que tiene la opción de generate getters and setters.

[07:04] Acá acemos un clic, en este caso vamos a estar haciendo un select getters, pero solamente de los tres campos que estamos utilizando. El de id todavía no lo vamos a utilizar entonces no lo vamos a generar. Ahí lo generamos, están todos los getters creados, guardé la clase y ya tenemos todo bien hecho acá también. Aquí podemos hacer una mejora más todavía.

[07:30] Mira una cosa. Nosotros acá en el método ejecuta registro, podríamos en lugar de estar pasando el nombre, la descripción y la cantidad, nosotros ya podríamos estar pasando directamente el valor de producto para guardar en la base de datos.

[07:48] Acá podemos seguir haciendo alguna mejoras, por ejemplo, el método acá de ejecutaRegistro, nosotros recibimos el producto, asignamos sus valores, sus atributos a variables y estamos enviando las variables para el método de ejecutaRegistro que realiza el insert en la base de datos.

[08:07] Lo que podemos hacer acá es, en lugar de estar utilizando el nombre, descripción y cantidad para guardar, enviar directamente el producto para esta clase, para este método, perdón, y acá en este método nosotros hacemos un Producto producto.

[08:27] Y en esos valores de setString, setString, setInt, nosotros utilizamos el producto.getNombre, producto.getDescripcion y producto.getCantidad. Ahí está. Nosotros podemos sacar esta lógica acá también del error. Podemos sacar. Para dejar el código un poco más sencillo, vamos a sacar esta lógica de valores máximos, para guardar en la base de datos, así nosotros podemos eliminar toda esta asignación de variables, para dejar más sencillo también.

[09:07] Entonces, ya aprendimos cómo funciona acá el control de transacciones, la vamos a seguir haciendo, pero la lógica ya no la tenemos más. Así estamos enviando el producto, recibimos acá la representación de un producto, un objeto. Lo están enviando para ejecutar el registro y acá solamente estamos utilizando sus valores, sus campos, para poder ejecutar de hecho la query de insert en la base de datos.

[09:38] Por fin, ¿qué vamos a estar haciendo? Acá en este resultSet, en lugar de estar imprimiendo el resultado acá, nosotros vamos a tomar este id que fue generado y lo vamos a asignar al producto. Entonces el producto.setId va a recibir este valor. Nosotros vamos a generar este método acá, el id.

[10.06] Vamos a hacer this.id = id; y acá en la impresión, en el system.out.println nosotros vamos a hacer lo siguiente. Voy a insertar el producto y esto vamos a cambiar para %s de string y vamos a arreglar el producto acá. Pero si hacemos esta forma, el toString del producto ¿va a estar haciendo qué? Va a imprimir la referencia del objeto en la memoria.

[10:37] Entonces nosotros tenemos que acá en el método, en la clase de producto, sobrescribir el método toString. Entonces hacemos aquí toString y el Eclipse ya nos ayuda con eso y hace la sobrescritura de este método para nosotros en lugar de llamar el super, nosotros vamos a hacer la siguiente lógica.

[11:00] String.format, vamos acá. Hacemos el format acá y vamos a dejarlo más o menos parecido con JSON. Va a hacer el "{id: %s, nombre: %s, descripción, %s, cantidad: %d}". Acá es una d porque es un valor numérico y ahí está. Cerramos la clave del JSON y ahora agregamos el this.id. Lo voy a poner más abajo para que podamos leer mejor.

[11:51] this.id, this.nombre, this.descripcion y this.cantidad; listo. Guardé los cambios y ahora vamos a ejecutar nuestra aplicación para hacer las pruebas de estos cambios. Bueno, ya se levantó la aplicación y acá voy a agregar un

producto nuevo que es un teclado. Y va a ser un teclado inalámbrico de 10 cantidades.

[12.21] Voy a guardar, ahí lo guardé, se registró. Mira, teclado acá de id 25, y tenemos acá en el log de la aplicación en la consola: Fue insertado el producto {id: 25, nombre: Teclado, descripción, Teclado Inalámbrico, cantidad: 10}. Ahí está, ahí me equivoqué. En lugar de poner dos puntos, puse coma, pero ahí están los valores todos bien. Listo.

[12:49] Ahora tenemos la representatividad del producto con una clase Java representando la tabla de la base de datos. Ahora tenemos que replicar esta solución para los demás métodos.

[13.00] Entonces acá nosotros vamos a tomar esta referencia de producto para hacer lo mismo con el listado, para el de eliminación y el de modificación también. Pero antes, veamos una cosa. En todos estos métodos que nosotros creamos para realizar alguna operación en la base de datos, nosotros tenemos que recuperar la conexión, crear un PreparedStatement para ejecutar la operación.

[13:28] Nosotros ya vimos esta situación anteriormente cuando creamos la connectionFactory para evitar la replicación del código. ¿Cómo hacemos para evitar esta situación que encontramos ahora? Debe haber alguna forma de extraer esta parte para un método o algo similar. Bueno, por ahora nosotros quedamos por aquí.

[13:47] En la próxima clase vamos a ver cómo resolver esta situación y hacer una buena refactorización en esta clase para seguir utilizando el modelo de producto.